Ordenação pelo Método da Intercalação

Merging Sort

- Princípio da Divisão e Conquista:
  - Divida o vetor A em dois subconjuntos A_1 e A_2
  - Solucione os sub-problemas associados a A_1 e A_2. Em outras palavras, ordene cada sub-conjunto separadamente (chamada recursiva)
    - Recursão pára quando atinge sub-problemas de tamanho 1
  - Combine as soluções de A_1 e A_2 em uma solução para A: intercale os dois sub-vetores A_1 e A_2 e obtenha o vetor ordenado
- Operação chave: **intercalação**.

- Procedimento intercala elementos nas posições p a q-1 com os elementos nas posições q a r, do vetor A.

```
procedure Intercala(p,q,r:integer; var A: vetor);
var i,j,k : integer;
temp: vetor;

begin
  i:= p;
  j:= q;
  k:= 1;

  while (i <= (q-1)) and (j <= r) do begin
     if (A[i] <= A[j]) then begin
        temp[k] := A[i];
        i:= i+1;
     end
     else begin
        temp[k] = A[j];
        j:= j+1;
     end;
     k:= k+1;
  end;
```

```
  while (i < q) do begin
     temp[k] := A[i];
     i:= i+1;
     k:= k+1;
  end;

  while (j <= r) do begin
     temp[k] := A[j];
     j:= j+1;
     k:= k+1;
  end;

end;
```

- Ordem de complexidade linear O(N)
- Precisa de vetor auxiliar.
- Para o caso de intercalação de listas lineares, o procedimento pode ser realizado com a mesma complexidade e sem necessidade de memória auxiliar, bastando a manipulação de apontadores.

- Procedimento MergeSort

```
procedure mSort (p, q:integer, var A: vetor)
var i: integer;
begin

   if (p < q) then begin
      i := (p + q) div 2;
      mSort(p,i,A);
      mSort(i+1,q,A);
      intercala(p,i+1,q,A);
   end;
end;


procedure MergeSort(var A:vetor; n:integer);
begin

   mSort(1,n,A);
end;
```

- Ordem de Complexidade O(n logn)

Ordenação pelo Método de Contagem

Distribution Counting

- Problema: Ordene um arquivo de N registros cujas chaves são inteiros *distintos* entre 1 e N

- Você poderia utilizar qualquer um dos métodos utilizados, mas deve tirar proveito das características específicas do arquivo (*chaves distintas entre 1 e N*)

- Solução:

```
for i:= 1 to N do t[A[i]] := A[i];
for i:= 1 to N do a[i] := t[i];
```

- Utiliza vetor auxiliar t de tamanho N.

- Ordem de complexidade linear O(N)

- Problema2: Ordene um arquivo de N registros, cujas chaves são inteiros entre 0 e M-1

- Idéia:

  - Conte op número de chaves com cada valor

  - Acumule os valores para determinar o número de chaves com valores menor ou igual a um determinado valor

  - Utilize estes contadores para mover os registros para as posições corretas, em um segundo passo

- Solução:

```
for j:= 0 to M-1 do
  count[j] := 0;

for i:=1 to N do
  count[A[i]] := count[A[i]] + 1;

for j:=1 to M-1 do
  count[j] := count[j] + count[j-1];

for i:=N downto 1 do
begin
     t[count[A[i]] := A[i];
     count[A[i]] := count[A[i]] - 1;
end;

for i:= 1 to N do A[i]:= t[i];
```

- Algoritmo eficiente se M não for muito grande.

- Complexidade linear O(max(N,M))

- Base para algoritmos *radix sortings*