Algoritmos de ordenação

Ordenação rápida ("Quicksort")

- Baseia-se num princípio muito simples que, quando aplicado recursivamente, acaba por ordenar o vector.
- Este princípio é composto por 2 passos essenciais:
  1. Escolher um elemento do vector, o pivot (por ex., V[0]);
  2. Dividir o vector em 2 partes (esquerda e direita), em que:
     - ✔ Parte esquerda com os elementos menores do que o pivot;
     - ✔ Parte direita com os elementos maiores do que o pivot.
- Ao ser aplicado succesivamente este princípio a ambas as partes, quando o processo recursivo terminar o vector está ordenado.

Algoritmos de ordenação

Ordenação rápida ("Quicksort")

O algoritmo é composto então por 3 passos:

1. Determinar a posição final k do elemento pivot (V[0]) no vector e colocá-lo nessa posição, dividindo o vector em 2 partes:
   - ✔ Esquerda = V[0], ..., V[k-1]
   - ✔ Direita = V[k+1], ..., V[tam-1]
2. Aplicar o algoritmo recursivamente à parte esquerda;
3. Aplicar o algoritmo recursivamente à parte direita.

Algoritmos de ordenação

Quicksort (Algoritmo)

Ordenar por Quicksort um subvector de *V* desde *inicio* até *fim*

Se (inicio < fim) então

k ← posição final do pivot, V[inicio], no vector

Ordenar por Quicksort o subvector esquerdo ao pivot

Ordenar por Quicksort o subvector direito ao pivot

Note-se que se (inicio >= fim) então o subvector de V tem apenas um elemento (inicio = fim) ou está vazio (inicio > fim). Logo, este subvector está ordenado.

Algoritmos de ordenação

Quicksort (Algoritmo)

Determinar a posição final *k* do pivot (V[inicio]) no subvector de V desde inicio até fim

```
k ← inicio
Para i desde (inicio+1) até fim fazer
    Se (V[i] < V[inicio]) então
        k ← k + 1
        Trocar (V[k], V[i])
Trocar (V[k], V[inicio])
Devolver (k)
```

Algoritmos de ordenação

Quicksort (função C)

```
void  Trocar  (int  *a, int  *b)
{
   int  aux;

   aux = *a;
   *a = *b;
   *b = aux;
}
```

Algoritmos de ordenação

Quicksort (função C)

```
int  DeterminarPivot  (int  V[],  int  inicio, int  fim)
{
   int  i, k = inicio;     //  k = posição do pivot V[inicio]
   for (i = inicio+1; i <= fim; i++)
      if (V[i] < V[inicio])
      {
         k++;
         Trocar (&V[i], &V[k]);
      }
   Trocar (&V[inicio], &V[k]);
   return (k);
}
```

Algoritmos de ordenação

Quicksort (função C)

```
void OrdenarQuicksort (int V[], int inicio, int fim)
{
   int k;    //  k = posição do pivot V[inicio]
   if (inicio < fim)
   {
      k = DeterminarPivot (V, inicio, fim);
      OrdenarQuicksort (V, inicio, k-1);
      OrdenarQuicksort (V, k+1, fim);
   }
}
```

Algoritmos de ordenação

Ordenação por fusão ("Merge Sort")

➔ A ordenação por fusão consiste em

- ✔ dividir o vector em vários subvectores de menor dimensão,
- ✔ ordenar estes subvectores separadamente e,
- ✔ fundi-los num vector único.

➔ O algoritmo de ordenação por fusão é recursivo e consiste em

- ✔ dividir sucessivamente o vector ao meio até que se obtenham subvectores com apenas 1 elemento (que está ordenado)
- ✔ fundir os subvectores num vector único.

Algoritmos de ordenação

Ordenação por fusão ("Merge Sort")

Pretende-se que o algoritmo de ordenação por fusão :

1. Ordene um vector V entre as posições a e b (V[a], V[a+1], ..., V[b]) e b+1 e c (V[b+1], V[b+2], ..., V[c]);
2. Fundir um vector V entre as posições a e c, tendo em conta que está ordenado entre as posições a e b (V[a] ≤ V[a+1] ≤ ... ≤ V[b]) e b+1 e c (V[b+1] ≤ V[b+2] ≤ ... ≤ V[c]).

Algoritmos de ordenação

Ordenação por fusão ("Merge Sort")

Ordenar por fusão um vector* V *entre as posições inicio e fim

Se (inicio < fim) então

meio ← (inicio + fim) / 2

Ordenar por fusão o vector V entre as posições inicio e meio

Ordenar por fusão o vector V entre as posições meio+1 e fim

Fundir os subvectores de V entre [inicio, meio] e [meio+1, fim]

Note-se que se (inicio ≥ fim) então o vector é composto apenas por um elemento (inicia = fim) ou está vazio (inicio > fim). Logo, está ordenado.

Algoritmos de ordenação

Ordenação por fusão (função C)

```
void  OrdenarFusao  (int  V[], int  inicio,  int  fim)
{
   int  meio;
   if (inicio < fim)
   {
      meio = (inicio + fim) / 2;
      OrdenarFusao (V, inicio, meio);
      OrdenarFusao (V, meio+1, fim);
      Fusao (V, inicio, meio, fim);
   }
}
```

Algoritmos de ordenação

Ordenação por fusão (função C)

```
void  Fusao  (int  V[], int  inicio, int  meio, int  fim)
{
   int  esq = inicio, dir = meio+1, k = 0, Aux[fim-inicio+1];
   while ((esq <= meio) && (dir <= fim))
   {
      if (V[esq] < V[dir]) {
         Aux[k] = V[esq];
         esq++;
      }
      else {
         Aux[k] = V[dir];
         dir++;
      }
      k++;
   }
```

Algoritmos de ordenação

Ordenação por fusão (função C)

```
   if (esq > meio)
      for (i = dir; i <= fim; i++)
         { Aux[k] = V[i]; k++; }
   else  // dir > fim
      for (i = esq; i <= meio; i++)
         { Aux[k] = V[i]; k++; }
   //  passar o vector Aux ordenado para o V
   for (i = 0; i < k; i++)
      V[inicio+i] = Aux[i];
}
```