QuickSort

O que é ?
O algoritmo de ordenação mais rápido para a maioria dos casos.

Como funciona?
- Divide o array em duas partes.
- Organiza cada uma delas, separadamente.

Como separar as partes?
- Escolher um pivô (pode ser qualquer um).
  IMPORTANTE: o pivô guarda um elemento.
- Os termos da esquerda serão menores ou iguais ao pivô e os termos da direita, maiores ou iguais.

Como fazer isso?
1ª etapa:
- Criar uma variável “dir” que vai receber a primeira posição do array.
- Criar uma variável “i” que recebe “dir” e percorre o array.
- Criar uma variável “esq” que recebe a última posição do array.
- Criar uma variável “j” que recebe “esq” e também vai percorrer o array.

2ª etapa:
- Percorrer a parte da direita do array enquanto o elemento da posição “i” for menor que o pivô.
- Percorrer o array da esquerda enquanto o elemento da posição “j” for maior que o pivô.
- Se a posição “j” for maior que a posição “i”, trocamos os elementos. É acrescido uma unidade em “i” e “j” é decrementado em uma unidade.
- Esses 3 passos vão se repetir enquanto “i” for menor ou igual a “j”.
Perceba então que vai acabar quando “j” estiver uma posição antes de “i”.

O que temos agora?
Dois novos arrays ! O primeiro com todos os elementos menores/iguais ao pivô e o segundo com todos os maiores/iguais.

E daí?

E daí que estamos diminuindo o nosso problema.
Lembra que a organização dos arrays é feita separadamente? Então vamos chamar essa função novamente mandando (vetor, esq, j) para ordenar a parte da esquerda e (vetor, i, dir) para ordenar a parte da direita.
Lembre-se de antes de chamar a função mandando esses argumentos, verificar >>separadamente<< se “j<esq” e se “i<dir”.
Perceber que agora “j” é “a nova direita” do array e “i”, a “nova esquerda”.

Agora voltamos para o início ! Um novo pivô será selecionada para cada array, e todas as etapas serão repetidas até que o array acabe.

OBS: a função recebe 3 argumentos: o vetor que deve ordenar, a posição da direita e da esquerda.